Docket No. RSW920030169US1

PARALLEL PROXIES

BACKGROUND OF THE INVENTION

1. Technical Field:

The invention relates to the network data processing field and, in particular, but not exclusively, to server performance in network data processing systems. Still more particularly, the present invention provides a method and apparatus for improving the performance of proxy servers in network data processing systems.

2. Description of Related Art:

A proxy is a server that typically resides between a client's application (e.g., Web browser) and another, "real" server in a network data processing system. A Web browser is a software application that can be used to locate and display documents stored on the World Wide Web ("Web") or the Internet in "Web servers". Web servers can be used to store and disseminate "Web pages". A Web browser (or similar client application) typically runs on a Personal Computer (PC) or workstation, and relies on the real server and/or one or more proxy servers to perform the Web browser's (or client application's) functions, such as, for example, locating and retrieving Web pages for display.

Proxy servers can be used to control access to Internet sites and provide certain Internet services. For example, proxy servers can be used to provide access to the Web or email messaging. A proxy configured to run

the known HyperText Transfer Protocol (HTTP) can be used to access the Web, and a proxy configured to run the known Simple Mail Transfer Protocol (SMTP) can be used for sending and receiving email. Also, proxy servers can be used for caching or storing Web pages, so that a subsequent request by a Web browser for a particular Web page can be satisfied locally from the proxy server, instead of routing the request back through the Web.

Figures 1A-1B depict a pictorial representation of a conventional, prior art network data processing system. Referring to **Figure 1A**, network data processing system **100** contains a PC or workstation **102**, which is configured with appropriate software to function as a Web browser within network data processing system **100**. As such, browser **102** can be connected via a conventional telecommunication network such as Local Area Network or Wide Area Network (LAN/WAN) **104** to a forward proxy server **106**. In the configuration shown, forward proxy server **106** may also be referred to as a "frontend" server, because forward proxy server **106** is located on the frontend or client side of network **108**.

As indicated by the dashed outline for forward proxy server **106**, network data processing system **100** can be alternatively configured to exclude forward proxy server **106**, and browser **102** can be connected directly to network **108** (e.g., via telecommunication network **104**). In this regard, a reverse proxy server **110** can be connected to network **108**. In the configuration shown, reverse proxy server **110** may also be referred to as a "backend" server, because reverse proxy server **110** is located on the

"backend" or server side of network **108**. The network configuration of a reverse proxy server as a backend Web server, as shown in **Figure 1A**, is often referred to as "intelligent routing" with respect to backend Web servers.

Reverse proxy server **110** is connected to a plurality of processing machines (e.g., **112**, **114**, **116**), which can be configured as servers or processors (e.g., Central Processing Units or CPUs). If browser **102** requests a document (e.g., Web page), the request is communicated to reverse proxy server **110** via networks **104** and **108**. Reverse proxy server **110** can then satisfy that request by retrieving the requested document from a processing machine **112**, **114** or **116**. Reverse proxy server **110** retrieves the requested document by addressing a Uniform Resource Locator (URL) associated with the document request received from browser **102**. The URL associated with the document request is addressed to the appropriate machine **112**, **114** or **116** where the requested document is stored. In this regard, each such URL can represent the global address of a document or associated processing machine **112**, **114**, **116** on the Web.

A significant problem that arises with the use of proxy servers in conventional network data processing systems is illustrated by **Figure 1B**. Typically, in conventional network data processing systems, one proxy server **110** is connected to a plurality of physical processing machines **112**, **114**, **116**. In order to process the document retrieval requests made by one or more browsers (e.g., **102**), proxy server **110** processes each

request for service in the order it is received. As shown in **Figure 1B**, the response time required for the proxy server **110** to forward these requests for service by the appropriate machine **112**, **114** or **116**, is the same duration (e.g., 10ms) for each such request made. In other words, proxy server **110** handles all of the requests on an equal basis. Consequently, in order to improve or decrease the response times for each of the requests, the conventional data processing network solution is to increase the ratio of proxy servers to processing machines (e.g., 3 proxy servers to 3 processing machines) so that each document request may be processed individually. However, this solution is costly in terms of hardware, processing time and money.

Therefore, it would be advantageous to provide an improved method, apparatus and program for increasing the performance of individual proxy servers with respect to the processing of document requests in network data processing systems, such as the Internet. In this regard, the conventional techniques being used to handle network traffic and make intelligent network handling decisions based on traffic content currently require the use of processing capabilities that impact the overall performance of the proxy servers involved. Consequently, there is a need to develop software solutions that can leverage the use of unique server designs in order to improve the performance of these proxy servers.

SUMMARY OF THE INVENTION

The present invention provides a plurality of prioritized proxies for processing service requests. In a preferred embodiment, a reverse proxy server is provided that can include 1-to-n proxy subunits configured in parallel, where "n" can vary depending on the total priority levels available for any given system. Each service (e.g., document-handling) request made to the proxy server is prioritized according to the prioritized proxy subunit that is configured to service the request. This prioritized proxy servers scheme increases the request handling response time for those requests being serviced by the higher priority proxy subunits, and decreases the response time for those requests being serviced by the lower priority proxy subunits. In this manner, the proxy server can significantly improve its overall request handling performance as compared to conventional proxy server request handling techniques.

BRIEF DESCRIPTION OF THE DRAWINGS

The novel features believed characteristic of the invention are set forth in the appended claims. The invention itself, however, as well as a preferred mode of use, further objectives and advantages thereof, will best be understood by reference to the following detailed description of an illustrative embodiment when read in conjunction with the accompanying drawings, wherein:

Figures 1A-1B are related drawings that depict a pictorial representation of a conventional, prior art network data processing system;

Figures 2A-2B are related drawings that depict a pictorial representation of a network data processing system in which the present invention may be implemented;

Figure 3 depicts a block diagram of a data processing system that may be implemented as a server, such as proxy server **206** in **Figures 2A-2B**, in accordance with a preferred embodiment of the present invention; and

Figure 4 depicts a block diagram of a data processing system that may be implemented as a client processor or browser, such as, for example, a client processor and/or browser **202a-202c** in **Figures 2A-2B**, in accordance with a preferred embodiment of the present invention.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

With reference now to the figures, **Figures 2A-2B** depict a pictorial representation of a network data processing system in which the present invention may be implemented. Network data processing system **200** is a network of computers, processors or servers in which the present invention may be implemented. Referring to **Figure 2A**, network data processing system **200** includes a plurality of client processors **202a-202c**, each of which can be a PC, workstation, server or other appropriate, digital processing machine.

In an exemplary embodiment, each client processor **202a-202c** is configured with appropriate software to function as a Web browser within network data processing system **200**. For example, each client processor **202a-202c** may be configured to function as a Web browser with such known browser software as Internet Explorer® or Netscape Navigator® running under an appropriate Operating System (OS), such as a Microsoft Windows® OS.

In this exemplary embodiment, each browser **202a-202c** can be connected to a network **204**, which is a medium used to provide communication links between various devices and computers connected together within network data processing system **200**. Network **204** may include connections, such as wire, wireless communication links, or fiber optic cables. In the depicted example, network data processing system **200** is the Internet with network **204** representing a worldwide collection of networks and gateways that use the Transmission Control

Protocol/Internet Protocol (TCP/IP) suite of protocols to communicate with one another. At the heart of the Internet is a backbone of high-speed data communication lines between major nodes or host computers, including thousands of commercial, government, educational and other computer systems that route data and messages. Of course, network data processing system **200** also may be implemented as a number of different types of networks, such as for example, an intranet, a local area network (LAN), or a wide area network (WAN). Also, network data processing system **200** may include additional servers, clients, and other devices not shown. **Figure 2A** is intended as an example, and not as an architectural limitation for the present invention.

For this exemplary embodiment, a proxy server **206** is preferably configured as a reverse proxy server and connected to network **204** on the backend side of network **204**. Proxy server **206** includes a plurality of proxy subunits **208a-208n**. Each proxy subunit of the plurality of proxy subunits **208a-208n** can be implemented with hardware and/or software to function as a separate proxy for receiving and processing requests (e.g., for documents) made by one or more browsers of the plurality of client processors or browsers **202a-202c**. Each proxy subunit **208a-208n** can be assigned a relative priority for processing such received requests.

For this exemplary embodiment, each of proxy subunits **208a-208n** can be connected to, and request documents or services from, a corresponding processing machine **210a-210n** based on software configuration. For example, with **n**

equal to **3**, each proxy subunit **208a-208c** is configured to send requests to a corresponding processing machine **210a-210c**. Each of processing machines **210a-210n** may be, for example, a server, network computer or PC.

Preferably, the level of priority assigned to a particular proxy subunit **208a-208n** is associated with the particular processing machine **210a-210n** configured to that proxy subunit **208a-208n**. For example, as shown in **Figure 2A**, a priority level of "1" can be assigned to proxy subunit **208a** for requests to be serviced by processing machine **210a**, a priority level of "2" can be assigned to proxy subunit **208b** for requests to be serviced by processing machine **210c**, and a priority level of "3" can be assigned to proxy subunit **208c** for requests to be serviced by processing machine **210b**.

In operation, for this exemplary embodiment, it may be assumed that one of the browsers **202a-202c** (e.g., browser **202a**) transmits a document request (e.g., for a Web page). The document request can be communicated to proxy server **206** by network **204**. Each proxy subunit **208a-208n** is functionally capable of satisfying such a request and retrieving the requested document from a particular processing machine **210a-210n**. As such, a specific proxy subunit **208a-208n** is assigned to satisfy that request by addressing a URL received from the browser (e.g., **202a**). The URL is associated with a particular processing machine **210a-210n** in which the requested document is stored. For this exemplary embodiment, the URL can represent the global address on the Web of the requested document, or the global address

of the processing machine **210a-210n** in which that document is stored.

Referring now to **Figure 2B**, for this exemplary embodiment, it may be assumed that the URL or document request received at proxy server **206** from the browser (e.g., **202a**) is for a document stored in processing machine **210a**. Also, it may be assumed that a priority level assigned to all document requests destined for processing machine **210a**, which requests are to be processed by proxy subunit **208a**, is a "high" priority relative to the priorities assigned to the other proxy subunits (e.g., **208b-208n**). As shown in **Figure 2B**, the received document requests with URLs destined for processing machine **210a**, which are to be handled by the "high" priority proxy subunit **208a**, are serviced by proxy subunit **208a** with the fastest response time (e.g., 5ms). Similarly, the received document requests with URLs destined for processing machine **210b**, which are to be handled by the "medium" priority proxy subunit **208b**, are serviced by proxy subunit **208b** with the second fastest response time (e.g., 8ms), and the received document requests with URLs destined for processing machine **210n**, which are to be handled by the "low" priority proxy subunit **208n**, are serviced by proxy subunit **208n** with the slowest response time (e.g., 17ms).

In the above-described manner, the present invention increases the document request handling response time for higher priority processing resources, and decreases the document handling response time for lower priority processing resources. Thus, the proxy server is free to

handle the distribution of a relatively large number of service requests. Also, the use of a plurality of parallel, prioritized proxies contained in an individual proxy server alleviates the bottlenecks that previously occurred during the handling of service requests by conventional proxy servers. As such, the present invention significantly improves the overall service request handling performance of a proxy server in a network data processing system (e.g., the Internet), as compared to conventional proxy server request handling techniques.

Advantageously, the present invention gives network data processing system designers greater control over the configuration of proxy servers, which allows network system administrators to configure a plurality of functional proxies on individual proxy server machines. As a result, the system administrators can prioritize the multiple, functional proxies to maximize proxy server performance, and also make informed decisions about the routing of traffic between multiple proxies, the proxy server, and the target processing machines. These decisions can be based on such network design and administrative factors as traffic content, the length of the content, the type of service request being made of the proxy server, the source and/or client applications that have made such requests, and the target processors that are destined to service such requests. Also, in accordance with the present invention, the improved design capability of encapsulating networking functions on a per-proxy, functional basis improves system

administrators' control over important functional components such as auditing and security, by enabling each proxy to provide a functional specialization.

Those of ordinary skill in the art will appreciate that the hardware depicted in **Figures 2A-2B** may vary. For example, other servers or similar processors, such as forward servers and the like, also may be used in addition to or in place of the hardware depicted. The depicted example is not meant to imply architectural limitations with respect to the present invention.

Referring to **Figure 3**, a block diagram of a data processing system that may be implemented as a server, such as proxy server **206** in **Figures 2A-2B**, is depicted in accordance with a preferred embodiment of the present invention. Data processing system **300** may be a symmetric multiprocessor (SMP) system including a plurality of processors **302** and **304** connected to system bus **306**. Alternatively, a single processor system may be employed. Also connected to system bus **306** is memory controller/cache **308**, which provides an interface to local memory **309**. An I/O bus bridge **310** is connected to system bus **306** and provides an interface to I/O bus **312**. Memory controller/cache **308** and I/O bus bridge **310** may be integrated as depicted.

Peripheral component interconnect (PCI) bus bridge **314** connected to I/O bus **312** provides an interface to PCI local bus **316**. A number of modems may be connected to PCI local bus **316**. Typical PCI bus implementations will support four PCI expansion slots or add-in connectors. Communication links to client processors (e.g., browsers)

202a-202c in **Figures 2A-2B** may be provided through modem **318** and network adapter **320** connected to PCI local bus **316** through add-in boards.

Additional PCI bus bridges **322** and **324** provide interfaces for additional PCI local buses **326** and **328**, from which additional modems or network adapters may be supported. In this manner, data processing system **300** allows connections to multiple network computers. A memory-mapped graphics adapter **330** and hard disk **332** may also be connected to I/O bus **312** as depicted, either directly or indirectly.

Those of ordinary skill in the art will appreciate that the hardware depicted in **Figure 3** may vary. For example, other peripheral devices, such as optical disk drives and the like, also may be used in addition to or in place of the hardware depicted. The depicted example is not meant to imply architectural limitations with respect to the present invention.

The data processing system depicted in **Figure 3** may be, for example, an IBM eServer pSeries system, a product of International Business Machines Corporation in Armonk, New York, running the Advanced Interactive Executive (AIX) operating system or LINUX operating system.

With reference now to **Figure 4**, a block diagram illustrating a data processing system is depicted in which the present invention may be implemented. Data processing system **400** is an example of a client processor, such as, for example, a client processor and/or browser **202a-202c** in **Figures 2A-2B**. Data processing system **400** employs a peripheral component interconnect (PCI) local bus

architecture. Although the depicted example employs a PCI bus, other bus architectures such as Accelerated Graphics Port (AGP) and Industry Standard Architecture (ISA) may be used. Processor **402** and main memory **404** are connected to PCI local bus **406** through PCI bridge **408**. PCI bridge **408** also may include an integrated memory controller and cache memory for processor **402**. Additional connections to PCI local bus **406** may be made through direct component interconnection or through add-in boards. In the depicted example, local area network (LAN) adapter **410**, SCSI host bus adapter **412**, and expansion bus interface **414** are connected to PCI local bus **406** by direct component connection. In contrast, audio adapter **416**, graphics adapter **418**, and audio/video adapter **419** are connected to PCI local bus **406** by add-in boards inserted into expansion slots. Expansion bus interface **414** provides a connection for a keyboard and mouse adapter **420**, modem **422**, and additional memory **424**. Small computer system interface (SCSI) host bus adapter **412** provides a connection for hard disk drive **426**, tape drive **428**, and CD-ROM drive **430**.

An OS runs on processor **402** and is used to coordinate and provide control of various components within data processing system **400** in **Figure 4**. The OS may be a commercially available OS, such as Windows XP, which is available from Microsoft Corporation. A browser-oriented programming system such as Microsoft's Internet Explorer® or Netscape's Navigator® may run in conjunction with the OS and provide calls to the OS from programs or applications executing on data processing system **400**. Instructions for the OS, the browser-oriented programming

system, and applications or programs are located on storage devices, such as hard disk drive **426**, and may be loaded into main memory **404** for execution by processor **402**.

Those of ordinary skill in the art will appreciate that the hardware in **Figure 4** may vary depending on the implementation. Other internal hardware or peripheral devices, such as flash read-only memory (ROM), equivalent nonvolatile memory, or optical disk drives and the like, may be used in addition to or in place of the hardware depicted in **Figure 4**. Also, the processes of the present invention may be applied to a multiprocessor data processing system. Also, data processing system **400** configured as a client processor and/or browser may relatively simply be a computer including a CPU, display monitor, and associated I/O and peripheral devices. As such, the depicted example in **Figure 4** and above-described examples are not meant to imply architectural limitations. For example, data processing system **400** also may be a notebook computer or hand-held computer in the form of a Personal Digital Assistant (PDA). Also, for example, data processing system **400** may be a kiosk, Web appliance, or Wireless-Fidelity (Wi-Fi) device.

It is important to note that while the present invention has been described in the context of a fully functioning data processing system, those of ordinary skill in the art will appreciate that the processes of the present invention are capable of being distributed in the form of a computer readable medium of instructions and a variety of forms and that the present invention

applies equally regardless of the particular type of signal bearing media actually used to carry out the distribution. Examples of computer readable media include recordable-type media, such as a floppy disk, a hard disk drive, a RAM, CD-ROMs, DVD-ROMs, and transmission-type media, such as digital and analog communications links, wired or wireless communications links using transmission forms, such as, for example, radio frequency and light wave transmissions. The computer readable media may take the form of coded formats that are decoded for actual use in a particular data processing system.

The description of the present invention has been presented for purposes of illustration and description, and is not intended to be exhaustive or limited to the invention in the form disclosed. Many modifications and variations will be apparent to those of ordinary skill in the art. The embodiment was chosen and described in order to best explain the principles of the invention, the practical application, and to enable others of ordinary skill in the art to understand the invention for various embodiments with various modifications as are suited to the particular use contemplated.